RSRE
MEMORANDUM No. 4197

ROYAL SIGNALS & RADAR ESTABLISHMENT

PARALLEL PROCESSING ON TRANSPUTER ARRAYS FOR THE RECOGNITION OF OBJECTS IN INFRA-RED IMAGES

Author: S A Baker

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
RSRE MALVERN,
WORCS.

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 4197

TITLE: PARALLEL PROCESSING ON TRANSPUTER ARRAYS FOR THE RECOGNITION OF OBJECTS IN INFRARED IMAGES

AUTHOR: S A Baker

DATE: September 1988

SUMMARY

This Memorandum describes work done in mapping image processing algorithms on to transputer arrays. Object classification based on the autocorrelation function has been applied to infrared images. Using the OCCAM programming language, parallel processing techniques have been developed to make use of the concurrency available for communication and calculation. The results for this specific problem are assessed and conclusions drawn on the use of transputers and OCCAM for parallel processing.

RSRE MEMORANDUM NO 4197

PARALLEL PROCESSING ON TRANSPUTER ARRAYS FOR THE RECOGNITION OF OBJECTS IN INFRA-RED IMAGES

S A Baker

CONTENTS

1 INTRODUCTION

Previous work [1] has shown that a simple measure of the shape of the autocorrelation function is a useful discriminant for target classification in infra-red images. It has been found that it is possible to recognise vehicles from texture information and, because object shape information is not used, it is well suited to thermal imagers which may not resolve shape or edge details. The algorithms in this paper have been adapted and developed for evaluation on arrays of transputers. Much experience has been gained in the design and development of algorithms and the approach required for efficient use of transputers.
The problem has been approached as follows :

i) Distribute the image under consideration across a network of transputers
ii) Reduce noise in the image
iii) Threshold the image to distinguish bright (hot) areas of potential interest
iv) Label connected regions and extract size and position parameters
v) Calculate the autocovariance
vi) Detect targets and display and file the results

At each stage care was taken to make as much use as possible of the concurrency available, and not to rely on the sequential algorithms already tested in [1]. One of the aspects of parallel processing investigated was the difficulties encountered in trying to adapt sequential algorithms for efficient execution on a parallel machine. In many cases a completely new approach is needed, and a simple transformation of the old code proves impossible. Conversely there are occasions when inherent parallelism in the problem can give excellent results with only minor changes to the sequential algorithm.

2 ARCHITECTURE

The Reconfigurable Transputer Processor (RTP) being developed in ESPRIT project P1085 uses hierarchically clustered T800 transputers connected through a VLSI switch. The T800 [2] is an advanced variant of the transputer developed for high speed manipulation of floating point numbers. The transputer is a single chip, powerful microprocessor designed as a module for use in the construction of multi-processor systems. The T800 incorporates a fast 32-bit integer processor and a floating point 64-bit processor, 4 kilobytes of fast static RAM and four high-speed communication links. Each RTP node ("Supernode") consists of 16 T800 worker transputers each with 256 Kbytes of external memory, a switch controller transputer, and an additional transputer with a large amount of memory (16 Mbytes) which can be used for storing and distributing data and code. All the transputer links are connected to a pair of VLSI switches, in such a way that a program running on the switch controlling transputer can set up any 4-connected network required. A single node can be used as a powerful workstation with a performance in excess of 16 Megaflops, or nodes may be connected together, again reconfigurably, to form a machine with up to 64 nodes.

3 TRANSPUTER ARRAY CONFIGURATION

The work was carried out on a prototype RTP node configured as shown in figure 1. T414 transputers (32-bit CPU and no floating point hardware) were used, and some facilities of the RTP machine, such as the command bus (mainly intended to aid diagnostic work) were not available.

The 16 'worker' transputers are connected in a 4 x 4 grid. Each processor works on a square patch representing 1/16 of the original image. The picture under consideration is sent from a file held on the disc of the IBM PC via the switch controller (SC) to processor 13 in the array. From there it is passed across and up the array (and up to a B007 transputer graphics board for visual display). It is intended in future to provide digitised images from a TV camera via a framestore controlled by a further transputer. The spare transputer links from processors P13 to P16 will be used to ease the bottleneck in distributing the picture

data to the array. It is clear from our results that this will lead to much enhanced over all throughput.

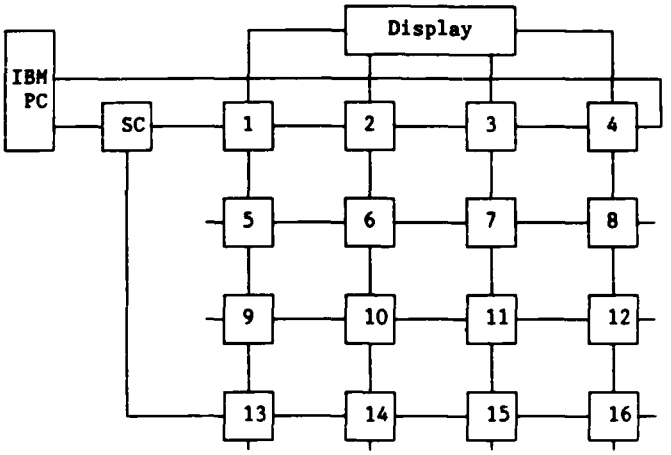

figure 1

4 OCCAM

Although transputer compilers for other languages (e.g. Prolog, C, Fortran) exist, OCCAM was chosen for programming this problem. OCCAM [3] is the first language to be based on the concept of parallel, in addition to sequential, execution, and to provide automatic communication and synchronisation between concurrent processes. OCCAM enables an application to be described as a collection of "processes", each executing concurrently, and able to communicate with other processes via "channels". The program is thus structured as modules, and the channels provide the connections between the modules. OCCAM thereby allows a program to be written in a clearly defined and structured manner, which exploits the parallelism available in a transputer network.

5 LOW LEVEL IMAGE PROCESSING

Several low-level operations are performed on the original image in order to derive a new image in which connected areas of a higher pixel intensity ("brightness") are distinguished from the surrounding areas of lower pixel intensity ("background").

a) CONVOLUTION

A convolution operation using a 3 x 3 weight matrix is first applied for 'noise' reduction. This smooths out isolated bright and dark pixels. The value of a pixel p at (i,j) is replaced by q where

$$q(i,j) = \left\{ \sum_{m=i-1}^{i+1} \sum_{n=j-1}^{j+1} p(m,n) \right\} / 8$$

It is clear that to perform this operation on pixels at the edge of a local image (i.e. at the edge of one of the distributed patches) pixel data is required from neighbouring processors. In each processor ar edge overlap is performed to extend the local picture by 1 pixel all round. First the horizontal and vertical edges are swapped and then the 4 corner pixels. For example in processor 6: first the edges are exchanged with the neighbouring processors 2, 7, 10 and 5, and then the corner pixels are exchanged with processors 1, 3, 11 and 9, via 2, 7, 10 and 5. The design of the transputer allows communication via any link to occur concurrently with communications on all other links, and so copying a small section of a neighbour's data is a very small overhead in overall processing time.

b) THRESHOLDING

The new filtered image is thresholded to produce a binary image. Several different approaches have been used to set the threshold :

i) a purely local threshold value based on a local (within the patch) maximum, minimum and mean (average) of pixel intensities.

ii) a global mean , in which case local values are sent to the controller and a global maximum, minimum and mean calculated. This gives poor results in pictures where areas are only 'bright' in a local context.

iii) a more 'graded' local threshold where local mean values for neighbouring processors are used in the calculation. The threshold value for any pixel depends on its relative position in the subimage i.e. its distance from any of the boundaries. This involves exchanging data between processors.

Method ii) was discarded as it gave poor results when the background across the whole image was not consistently darker than the possible targets. Method iii) was the most effective but took 50% longer to calculate, as it involved data swapping and a more complicated algorithm. In most cases, method i) would be adequate.

In each processor a binary image is created containing 1's where there are pixels above the threshold level in the filtered picture and 0's elsewhere. It is now assumed that the pixels in any possible areas of interest in the original image have been selected. The only pixels which will be considered are those marked by a 1 in the binary image.

6 TARGET DISCRIMINATION AGAINST BACKGROUND

'Targets' are connected areas of pixels which have been marked by a 1 in the binary image. The mark is changed to a target marker value. In the binary image, the neighbours of a point found to be in a target are checked for membership of the same target. When a new point is added to a target, the parameter values for that target (area, x and y minimum and maximum) are updated. If it is found that one target is 'touching' another i.e. adjacent pixels have different marker values, an entry is made in a merger table. When all the pixels in the binary image have been scanned, and all the 1's have been replaced by markers, the merger table is used to re-mark each target with a consistent marker value and to

merge the parameter values. At this stage each processor contains a list of 'targets' and a picture marked to show the pixels in each target. Only the actual points contained in each target are needed for the calculation of the autocorrelation function. This can lead to complications when the local target contains points along the edge of the local picture. Care must be taken not to introduce discontinuities across processor boundaries which would introduce errors into the calculations.

7 AUTOCORRELATION

The one-dimensional autocorrelation function $\phi(\tau)$ of $x(t)$, referred to a displacement τ is defined by

$$\phi(\tau) = \int_{-\infty}^{\infty} x(t).x(t+\tau)\, dt$$

For a discretely sampled x, the autocorrelation coefficient for a displacement r is estimated as

$$R_r = R(r) = \phi(r\Delta\tau) = \frac{1}{N-r} \sum_{n=1}^{N-r} (x_n - \bar{x}).(x_{n+r} - \bar{x})$$

where N is the number of samples used and $\bar{x}$ the mean value of x (which here corresponds to image intensity) within the marked target.
So that

$$R_0 = \frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})^2 = S^2 = \text{variance of } x$$

If R_r is normalised we obtain the autocovariance $C_r = R_r/R_0$.
Then

$$-1 < C_r < +1 \quad \text{and} \quad C_0 = 1$$

The shape parameters S_1 and S_2 approximate the magnitude and sense of curvature of the autocovariance near its peak, on two different scales of distance.

$$S_1 = (C_1 - C_2)/(C_0 - C_1)$$

$$S_2 = (C_2 - C_4)/(C_0 - C_2)$$

and we have used the values of C_1 and S_1 and S_2 to classify targets. Correlation functions in the x-direction only are used. This is for simplicity and in order to prove the concepts rather than to make a fully automatic classifier.

8 AUTOCOVARIANCE CALCULATION

The equation for C_r requires that the mean value $\bar{x}$ within each separate target needs to be calculated before accumulating sums of products. To avoid scanning the image twice , first to find the mean and secondly to calculate C_r for each target, the equation can be rewritten

$$Cr = \left[\frac{1.}{N_y} \sum_{1}^{N_y} \frac{1}{N_{x-r}} \left(\sum x_n . x_{n+r} - \bar{x} . \sum (x_n + x_{n+r}) + \bar{x}^2 \right) \right] / R_0$$

where

N_y = number of lines in target subimage
N_x = number of pixels on line within subimage
$= x_{max} - x_{min} + 1$
$r = 0,1,2,3,4$

In practice, there may be gaps along a line between pixels which belong to the same target, in which case each set of connected points is treated as a separate line, local values for x_{max} and x_{min} are used, and N_y is amended.

If any of the targets found and marked in the binary image is completely contained within its own processor, and does not include any points on the boundary of the local image, the autocovariance calculation can now be computed. In the tables giving timing details these targets are referred to as 'local targets'. Two different approaches to the

problem of targets which overlap adjacent processors ('global targets') have been tested, and a third considered.

i) The target parameters and the marked picture are sent up the array from each processor and collected by the transputer in the IBM PC. Touching targets are combined, and for each target the autocorrelation function can be calculated using the original picture stored on the IBM PC disc and an amalgamated marked picture.

ii) Partial sums of pixel values , products and number of valid pixels are amassed locally and sent to the IBM PC for collation (or any 'controller' transputer could be used). To allow values of r up to 4, where r is the distance between pixels in a sum or product in the equation, extra overlaps of original picture and marked target information are required from neighbouring transputers.

iii) A strategy of switching the machine connectivity was considered, in which processing is done as above up to the target discrimination stage, and then the transputer array is reconfigured to cope with the ACF calculation to make more efficient use of all the processors. Unfortunately it is not possible to reconfigure the transputer network during program execution without making use of the control bus, which was not available on the prototype machine. It is planned to examine the feasibility of this approach on the ESPRIT RTP node when it becomes available.

9 RESULTS SUMMARY

The variants of the above algorithms were programmed and the results compared. The figures for timings were obtained using many methods for attaining faster performance, in particular

a) an OCCAM compiler which optimises the use of the fast local memory

b) optimising techniques available in the OCCAM language

c) as much use of concurrency as possible, in particular with regard to overlapping calculations and communications.

The following execution times were achieved over the 16 transputers for convolution and thresholding (timings in seconds)

Processor	picture overlap	convolution	local threshold	graded threshold
P1	0.0019	0.4344	0.1292	0.1827
P2	0.003	0.4364	0.1293	0.1787
P3	0.0033	0.4364	0.1291	0.179
P4	0.0026	0.4364	0.1293	0.1789
P5	0.0026	0.4364	0.1292	0.1791
P6	0.0029	0.4364	0.1292	0.1791
P7	0.0031	0.4364	0.1292	0.1791
P8	0.0025	0.4364	0.1295	0.1788
P9	0.0026	0.4364	0.1295	0.1782
P10	0.0029	0.4364	0.1295	0.1782
P11	0.0029	0.4364	0.1295	0.1786
P12	0.0023	0.4364	0.1296	0.1784
P13	0.0019	0.4364	0.1295	0.1817
P14	0.0022	0.4364	0.1297	0.1778
P15	0.0022	0.4364	0.1296	0.178
P16	0.0022	0.4364	0.1294	0.1784

It is easy to see that apart from the overlap function, where position in the transputer network is important (for instance P1 has only 2 edges to swap), the low-level operations are data-independent and timing differences are caused by the positioning of variables in memory by the compiler or programmer. It is obviously important to make as much use as possible of the fast local memory.

For the autocovariance calculation, timings were obtained for both methods i) and ii) of Section 8 for targets which overlap processor boundaries.

i) Timings for the transputers in the network when target merging and calculations are done on the IBM PC (in seconds)

Processor	target discrimination	calculate local ACF	total run time
P1	0.4055	0.0	4.4127
P2	0.4255	0.0	4.4009
P3	0.5176	0.0	3.4486
P4	0.4611	0.1062	3.449
P5	0.3964	0.0178	4.3851
P6	0.3567	0.0	4.3599
P7	0.5514	0.0083	3.3889
P8	0.2952	0.0529	3.3892
P9	0.3178	0.0	4.4259
P10	0.3494	0.0	3.3467
P11	0.2079	0.0	3.3396
P12	0.2649	0.0	3.3366
P13	0.3681	0.0	5.7917
P14	0.2649	0.0	4.425
P15	0.2191	0.0164	3.4375
P16	0.8854	0.098	3.4378

and timings on the IBM PC transputer are :

global amalgamation and calculation : 1.4365s

total run time (including 6.015s to read file) : 16.2131s

ii) Timings in the network when partial calculations are done locally and amalgamated on the IBM PC transputer

Processor	target discrimination	global ACF calculation	total run time
P1	0.3975	0.1852	2.4842
P2	0.424	0.1202	2.4824
P3	0.5151	0.1731	2.4824
P4	0.4535	0.0	2.4827
P5	0.3952	0.1619	2.427
P6	0.3571	0.1496	2.4262
P7	0.5478	0.14	2.4265
P8	0.2948	0.0	2.4268
P9	0.3165	0.0049	2.3887
P10	0.3481	0.0259	2.251
P11	0.2085	0.0125	2.2511
P12	0.2646	0.0184	2.2512
P13	0.3617	0.0008	2.4326
P14	0.2644	0.0065	2.3879
P15	0.2195	0.0	2.2585
P16	0.8747	0.0	2.2588

with timings on the IBM PC transputer :

time to receive partial ACF elements : 0.1921s

time to collate partial sums : 0.0591s

total run time (including 6.0089s to read file) : 12.4522s

It can easily be seen that method ii), even though it involves extra data movement, is much faster (ignoring the file reading time, processing takes 6.443s rather than 10.198s).

10 CONCLUSIONS

Many conclusions can be drawn from the results of the various methods and techniques employed to calculate autocorrelation functions using a network of transputers. The problem has two distinct components - the low-level image processing, followed by the feature extraction and calculation phase. It is clear that during the low-level stage all the transputers in the network are doing the same operations at the same time and so there are no problems of efficiency, or need for load balancing. In fact this is true of any problem which can be decomposed in a geometric fashion into identical sub-problems working on a subset of the data with only a minimum need for inter-processor communication.

The second phase is much more data-dependent and so it is more difficult to design a problem-solving strategy which uses all the transputers as efficiently as possible. At first it was felt that the quasi-dynamic switching approach would be the most efficient. In this method, once target areas have been located in the subimages, the feature extraction and ACF calculation would be farmed out over a new network. However this could involve a large amount of data redistribution, not only of the original picture but also of the marked target areas. It is soon found that the addition of large amounts of data passing is both time consuming (at run time) and also presents many opportunities for introducing deadlock, a situation in which two or more processes are perpetually waiting for communication from each other.

The communications handling soon proved itself to be the more complicated task from the programmer's point of view, while the coding of the numerical algorithms was relatively easy (particularly with the checking mechanisms available in the OCCAM compiler). Any solution using parallel processing techniques needs to take into account how much programming time will be required, not only for designing and coding but also for testing and debugging. The debugging tools available at present are rather limited, although more aids are being developed. There is

often a heavy penalty to pay (in terms of programmer time) for sophisticated algorithms which produce enhanced performance. It is often extremely difficult to make even minor modifications, and in some cases a small change in specification can lead to a major redesign of the program. The importance of the algorithm design phase, in particular the choice of parallel decomposition of the problem, cannot be over-emphasised. A simplicity of concept is of paramount importance.

11 ACKNOWLEDGEMENTS

This work is partially funded under ESPRIT contract P1085. The ESPRIT programme (European Strategic Programme for Research and development in Information Technology) comprises precompetitive R & D projects, carried out by collaborative effort, on a shared cost basis. P1085 is a project in the Computer Architecture part of the Advanced Information Processing subprogramme. Its objective is to develop a low cost high performance multiprocessor computer with supporting software and a range of applications to demonstrate its performance. The collaborators in P1085 are RSRE (the prime contractor), APSIS, Inmos, Telmat S.A., Thorn-EMI, IMAG and Southampton University. Part of the work done at RSRE has been to develop applications in Image Processing and Target Recognition.

The author has pleasure in thanking those members of SP2 who have given advice on aspects of hardware, software, image processing and presentation. In particular KJ Palmer, HC Webber, Kevin Collins, JG Harp and JBG Roberts.

12 REFERENCES

1 Ph.D. Thesis "Algorithms and Architectures for Image Processing" University of Surrey JG Harp 1988

2 TRANSPUTER REFERENCE MANUAL INMOS Ltd PRENTICE HALL 1988

3 OCCAM 2 REFERENCE MANUAL INMOS Ltd PRENTICE HALL 1988

DOCUMENT CONTROL SHEET

Overall security classification of sheet ..... UNCLASSIFIED .....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification
	Memorandum 4197		Unclassified

5. Originator's Code (if known)	6. Originator (Corporate Author) Name and Location
7784000	Royal Signals and Radar Establishment St Andrews Road, Malvern, Worcestershire WR14 3PS
5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location

7. Title
PARALLEL PROCESSING ON TRANSPUTER ARRAYS FOR THE RECOGNITION OF OBJECTS IN INFRARED IMAGES

7a. Title in Foreign Language (in the case of translations)

7b. Presented at (for conference papers) Title, place and date of conference

8. Author 1 Surname, initials	9(a) Author 2	9(b) Authors 3,4...	10. Date	pp. ref.
Baker S A			9.88	14

11. Contract Number	12. Period	13. Project	14. Other Reference

15. Distribution statement
Unlimited

Descriptors (or keywords)

continue on separate piece of paper

Abstract

This Memorandum describes work done in mapping image processing algorithms on to transputer arrays. Object classification based on the autocorrelation function has been applied to infrared images. Using the OCCAM programming language, parallel processing techniques have been developed to make use of the concurrency available for communication and calculation. The results for this specific problem are assessed and conclusions drawn on the use of transputers and OCCAM for parallel processing.